NON-PROVISIONAL APPLICATION FOR UNITED STATES PATENT

FOR

PROCESSING BLOCK WITH INTEGRATED LIGHT WEIGHT MULTI-THREADING SUPPORT

Inventor
Mehta, Kalpesh D.

Prepared by: Schwabe, Williamson & Wyatt, PC
Pacwest Center
1211 SW Fifth Ave., Ste 1600-1900
Portland, Oregon 97204

Attorney Docket No.: 110349-133005
IPG No: P16599

Express Mail Label No. EL782197076US
Date of Deposit: August 29, 2003

BACKGROUND OF THE INVENTION

Numerous data processing applications require a relatively small number of unique operations to be repeatedly performed for a large volume of data. For example, in a number of media applications, such as processing of video data, a relatively small number of unique operations are repeatedly performed on many blocks of many frames/pictures of video data.

As integrated circuit technology continues to advance, it is desirable to have media processors that are custom designed for such type of processing. In particular, it is desirable to have media processors designed with multiple data processing blocks equipped to repeatedly perform these relatively small number of operations for the large volume of data, in a cooperative and at least partially parallel manner.

Further, it is desirable for each of the data processing blocks to operate with a high degree of efficiency. Thus, it is also desirable for the data processing blocks to be able to support multi-threading (interleaved execution of multiple threads of instructions), without the typical significant resource overhead required to support context switching (saving and restoring the various thread states as execution switches back and forth between the different threads of instructions).

BRIEF DESCRIPTION OF THE DRAWINGS

Embodiments of the present invention will be described by way of the accompanying drawings in which like references denote similar elements, and in which:

Figure 1 illustrates an overview of a processing block of the present invention, in accordance with one embodiment;

Figure 2 illustrates a number of example "simple" threads of **Fig. 1** in further details, in accordance with one embodiment;

Figure 3 illustrates the thread switching structure of **Fig. 1** in further details, in accordance with one embodiment;

Figures 4a-4c illustrate the relevant operational logic of the thread management unit of **Fig. 1**, in accordance with one embodiment;

Figure 5 illustrates a signal processing macroblock formed employing various variants of the processing block of **Fig. 1**, in accordance with one embodiment;

Figure 6 illustrates a digital media processor incorporated with a number of the signal processing macroblocks of **Fig. 5**, in accordance with one example application of the present invention; and

Figure 7 illustrates a digital system incorporated with the digital media processor of **Fig. 6**, in accordance with one example application of the present invention.

DETAILED DESCRIPTION OF EMBODIMENTS OF THE INVENTION

Embodiments of the present invention include but are not limited to processing blocks equipped with light-weight multi-threading support, signal processing macroblocks formed with such processing blocks, as well as media processors formed with such signal processing macroblocks.

In the following description, various aspects of embodiments of the present invention will be described. However, it will be apparent to those skilled in the art that other embodiments may be practiced with only some or all of the described aspects. For purposes of explanation, specific numbers, materials and configurations are set forth in order to provide a thorough understanding of the embodiments. However, it will be apparent to one skilled in the art that other embodiments may be practiced without the specific details. In other instances, well-known features are omitted or simplified in order not to obscure the description.

Various operations will be described as multiple discrete operations in turn, in a manner that is most helpful in understanding the embodiments, however, the order of description should not be construed as to imply that these operations are necessarily order dependent. In particular, these operations need not be performed in the order of presentation.

The phrase “in one embodiment” is used repeatedly. The phrase generally does not refer to the same embodiment, however, it may. The terms “comprising”, “having” and “including” are synonymous, unless the context dictates otherwise.

Referring now to **Fig. 1** wherein an overview of a processing block of the present invention, in accordance with one embodiment, is illustrated. As shown, for the embodiment, processing block **100**, coupled to external register set **110**, may include execution unit **102**, fetch and decode unit **103**, instruction memory **104**, thread management unit **105**, control memory **106**, and I/O interface **108**, coupled to each other as shown.

Instruction memory **104** may be employed to stage (store), various threads **122** of instructions to be executed interleavingly. Control memory **106** may be employed to store a relatively simple thread switching structure **124**, to allow thread management unit **105** to facilitate interleaved execution of threads **122** for processing block **100**, without requiring significant amount of resources to track the thread states (light-weight).

The resulting benefit is processing block **100** becoming highly efficient, executing different threads of instructions interleavingly, but remain compact and small. The efficiency and compactness makes processing block **100** particularly suitable for multiple replication to form a signal processing macroblock on an integrated circuit, which in turn, may be replicated multiple times on the same integrated circuit, to form a highly efficient and powerful single chip media processor.

In various embodiments, instruction memory **104** and control memory **106** may be static or dynamic random access memory (SRAM and DRAM). In other embodiments, instruction memory **104** and control memory **106** may be electrically eraseable read-only memory (EEPROM) or flash memory. In yet other embodiments, instruction memory **104** and control memory **106** may be memory of other known types or to be designed.

Fetch and decode unit **103** may be employed to fetch and decode instructions of the various threads **122**, under the control of thread management unit **105**. Except for the fact that fetch and decode unit **103** operates under the control of thread management unit **106**, fetch and decode unit **103** represents a broad range of such element known in the art or to be designed.

Execution unit **102** may be employed to execute instructions. Execution unit **102** may include an arithmetic logic unit (ALU). For the embodiment, execution unit **102** may further include support for notifying thread management unit **105** of execution of certain thread switching related instructions, to be described more fully below. Beyond that, the exact makeup of execution unit **102** may be embodiment dependent. For example, for computational intensive processing blocks, execution unit **102** may be endowed with multiplication units and/or floating point processing supports. Thus, execution unit **102** also represents a broad range of such elements known in the art or to be designed.

Thread management unit **105** may be employed to control the interleaved execution of threads **122**, including in particular the fetching and execution of a number of thread switching related instructions. In one embodiment, these thread switching related instructions may include a create thread instruction (crth), a thread termination instruction (kill), and a thread switching instruction (switch). The meaning, usage, and the operational logic of thread management unit **105** in support of these instructions, including the operations performed on thread switching structure **124** as part of the fetching and execution of these instructions, will be further described below.

Register set **110** may include a number of registers **112** to store operand data. For the embodiment, to facilitate cooperative processing, as described earlier, register set **110** may be an external set of registers, sharable by a number of processing blocks. In various embodiments, different non-overlapping subsets of registers **112** may be used by the different executing threads of a processing block. The practice reduces the need of resources to facilitate thread switching.

Further, for the embodiment, to facilitate cooperating processing by multiple processing blocks, each register **112** may include a number of data valid indicators **116**, in addition to data storage area **114**. Each data valid indicator **116** (set by the executing threads) indicates whether the data stored in data storage area **114** is valid for a corresponding processing block. Thus, thread management unit **105** of processing block **100** may determine whether a thread is ready for execution, based on the states of data valid indicators **116** corresponding to processing block **100** of the subset of registers **112** used by the thread.

However, alternate embodiments may be practiced without the coordinated practice of having the threads of a processing block employing different non-overlapping subsets of the external registers, provided corresponding resources required to save and restore the proper states of the overlapped registers are provided. Further, alternate embodiments may be practiced without thread readiness determination or with thread readiness being determined in other manners.

Similarly, while the embodiment is designed to allow multiple processing blocks **100** be employed to form a highly efficient processing macroblock, other embodiments may be practiced having only one processing block **100**.

In various embodiments, I/O interface **108** may be a configurable interface configurable to be either an input interface or an output interface. For the former case, processing block **100** effectively becomes an input processing block, whereas for the latter case, processing block **100** effectively becomes an output processing block. One example usage of these example input or output processing blocks will be described later.

Figure 2 illustrates a number of example threads of instructions, threads **122a-122e**, in accordance with one embodiment. Of particular interest, for the embodiment, are three thread switching related instructions **202-206**.

Instruction **202** is a create thread instruction, allowing a thread, e.g. thread0 **122a**, to spawn the execution of other threads, e.g. thread1-thread4 **122b-122e**. For the embodiment, thread instruction **202** may include specifications of the dependencies of the thread being spawn. These dependencies may include the subset of registers used by the thread being spawn. These dependencies may also include the I/O ports used by the thread being spawn.

Instruction **204** is a thread execution termination instruction, allowing a thread, e.g. thread0 **122a**, to terminate its own execution.

Instruction **206** is a thread execution switch instruction, allowing a thread, e.g. thread1-thread4 **122b-122e**, to cause execution to be switched to another thread.

As alluded to earlier, and to be described in more detail below, thread management unit **105** includes support for the execution of each of these three instructions.

Other illustrated instructions, “pack”, “shr”, “mpy” and “add” are representative of a broad range of “arithmetic” instructions that may be also supported by execution unit **102**. Example “pack” instruction represents a convention “pack” function instruction to perform a packing operation on its operands. Example "shr", "mpy" and "add" instructions represent convention "shift right", "multiply" and "add" arithmetic operation instructions to perform the corresponding convention shifting, multiplication and adding operation on the respective operands.

Figure 3 illustrates the thread switching structure of **Fig. 1** in further details, in accordance with one embodiment. As illustrated, for the embodiment, thread switching structure **124** may include a current thread identifier **302** to identify a current one among the plurality of threads **122** as the current thread being executed by execution unit **102**.

For the embodiment, thread switching structure **124** may further include thread array **304** for storing a number of thread entries **306**, one per thread, to correspondingly describe threads **122** being executed interleavingly.

For the embodiment, each thread entry **306** may include a thread program counter (PC) **308** to identify the next instruction of the corresponding thread to be executed (when the corresponding thread becomes the current thread to be executed).

Further, for the embodiment, each thread entry **306** may include an activeness indicator **310** indicating whether the corresponding thread is in an active state or in an inactive state. For the embodiment, the corresponding thread will be included among the thread to be considered for execution, when execution is being switched from one thread to another thread, if the activeness indicator **310** of the corresponding thread indicates an active state.

For the embodiment, each entry **306** may further include other information describing the corresponding thread **122**, e.g. the earlier described dependency information of the corresponding thread **122**, including but are not limited to the subset of the external registers, and the I/O ports used by the corresponding thread **122**.

Figures 4a-4c illustrate the relevant operational logic of thread management unit **105** in support of the earlier described instructions for facilitating light-weight multi-threading, in accordance with one embodiment.

As illustrated in **Fig. 4a**, on notification by execution unit **102** of the execution of the create thread instruction (crth), thread management unit **105** adds a thread entry **306** in thread array **304** for the thread being spawned for execution, block **402**. As described earlier, for various embodiments, the thread entry **306** may include a PC and an activeness indicator for the thread being spawned for execution.

As illustrated in **Fig. 4b**, for the embodiment, on notification by execution unit **102** of the execution of the thread execution termination instruction (kill), thread management unit **105** resets the activeness indicator of the thread entry **306** in thread array **304** for the thread, which execution is being terminated, block **412**.

As illustrated in **Fig. 4c**, for the embodiment, on notification by execution unit **102** of the execution of the thread execution switching instruction (swtch), thread management unit **105** selects another thread, among the active and ready ones of threads **122**, to be the current thread, block **422**. The selection may be made in any one of a number of manners, including but not limited to a round-robin based manner, a fixed priority based manner, and a rotating priority based manner.

On selecting another active and ready thread to be the current thread, thread management unit **105** updates the current thread identifier **302** to identify the selected thread as the current thread to be executed, block **424**.

Thereafter, thread management unit **105** instructs fetch and decode unit **103** to fetch and decode instructions for the now current thread, in accordance with the PC of the current thread.

Figure 5 illustrates a signal processing macroblock formed using multiple ones of the earlier described processing block, in accordance with one embodiment. As illustrated, signal processing macroblock **500** may include a number of variants of the processing block of **Fig. 1**, processing blocks **502a-502d**, register set **504**, hardware accelerator **506**, memory command handler **508**, and local memory **510**, coupled to each other as shown.

For the embodiment, processing blocks **502a-502b** are input and output processing blocks respectively, i.e. processing block **100** with input/output interface **108** configured as an input interface in the former case, and as an output interface in the latter case. Processing blocks **502c-502d**, on the other hand, are variants of the earlier described computational processing blocks.

Register set **504** may be a variant of register set **110** of **Fig. 1**, and local memory **510** may be any one of a number of memory known in the art or to be designed. Memory command handler **508** may be any one of such element known in the art or to be designed.

In various embodiments, hardware accelerator **506** may include an address generator equipped to generate access addresses for accessing a unit of data in a non-sequential access manner, e.g. a zig-zag pattern. The address generator is the subject matter of co-pending application number <to be

inserted>, entitled "Non-sequential Access Pattern Based Address Generator", contemporaneously filed with the present application.

During operation, data to be processed are inputted in the signal processing macroblock **500**, more particularly, registers of register set **504** and/or local memory **510**, through input processing block **502a**. The inputted data, in turn, are processed by selected ones of processing blocks **502a-502d**, with the processing results being placed back in the registers of register set **504** and/or local memory **510**. The processing results, in due course, are outputted from the registers of register set **504** and/or local memory **510**, through output processing block **502b**.

Figure 6 illustrates a digital media processor incorporated with the teachings of the present invention, in accordance with one embodiment. As illustrated, digital media processor **600** may include a number of signal processors (SP) **602**, and a number of direct memory access (DMA) units **604**, coupled to each other as shown.

SP **602** may be equipped to cooperate with each other to process digital media data. In various embodiments, one or more of SP **602** may be variants of signal processor **500** of **Fig. 5**.

DMA units **604** may be equipped to retrieve the digital media data from external memory for SP **602**.

In one embodiment, the above described digital media processor **600** may be disposed in a single integrated circuit.

Figure 7 illustrates a digital system incorporated with the teachings of the present invention, in accordance with one embodiment. Digital system **700** may include digital media processor **600** of **Fig. 6**, DDR memory **702**, host processor **704**, memory **706** and bus **708** coupled to each other as shown.

In other words, one or more signal processors of the digital media processor **600** may be equipped with processing blocks incorporated with the earlier described light weight multi-threading support.

Otherwise, DDR memory **702**, memory **706**, host processor **704** and bus **708** all represent a broad range of these elements known in the art or to be designed.

In various embodiments, digital system **700** may be a server, a desktop computer, a laptop computer, a tablet computer, a pocket PC, a palm sized personal digital assistant, a wireless mobile phone, a set-top box, an entertainment control console, a video recorder, or a video player.

Thus, it can be seen from the above descriptions, a novel processing block and a number of its example applications have been described.

While the present invention has been described in terms of the foregoing embodiments, those skilled in the art will recognize that the invention is not limited to the embodiments described. Other embodiments may be practiced with modification and alteration within the spirit and scope of the appended claims.

Thus, the description is to be regarded as illustrative instead of restrictive.